Flowers-102 Classification Model

Y3918764

***Abstract*—This report details the development of a Convolutional Neural Network (CNN) for classifying 102 different flower species in the Oxford Flowers-102 dataset. The network comprises six convolutional layers, followed by max-pooling layers and a fully connected layer. Regularisation techniques batch normalisation and dropout were applied after each convolutional layer to prevent overfitting. Pre-processing methods, such as cropping, resizing, and data augmentation, were utilised to improve model performance given the limited training data. Hyperparameter tuning was essential for optimisation, with the final configuration using a low learning rate and the Adam optimiser with a learning rate scheduler. The final model achieved an accuracy of 68.48% on the test set.**

I. Introduction

IMAGE classification, a fundamental task in machine learning, involves training models to categorise images based on visual features. The new technology is crucial for applications such as enhancing diagnostic accuracy in medical imaging and optimising crop surveillance in agriculture [1]. Specifically, classifying flower species is integral to botany, agriculture, and environmental conservation, aiding in biodiversity monitoring and habitat assessment [2]. Traditionally, image classification relied on manual feature extraction methods like the Scale-Invariant Feature Transform (SIFT) [3], which generates descriptors invariant to scale and orientation changes. However, Convolutional Neural Networks (CNNs) have revolutionised this field by automating feature extraction and learning hierarchical visual features directly from data [4]. Originating from early neural network models [5], CNNs are adept at handling complex visual tasks through layers of processing. Despite their capabilities, CNNs face challenges, especially with limited datasets and intricate class differentiations. This project develops a neural network model for classifying flower species from the Flowers-102 dataset [6]. Given the dataset's limited size, the study emphasises architecture design and hyperparameter tuning to achieve effective classification results. These decisions are crucial in enhancing model performance, particularly when extensive data is unavailable. Our approach demonstrates significant improvements through meticulous model design and innovative data augmentation techniques, contributing valuable insights into flower species classification.

II. Method

A. Neural Network Architecture

The final network consists of a multi-layer architecture designed to effectively classify the 102 flower species as shown in Figure 1. Each layer serves a specific purpose to enhance the model's performance. Initially, an input layer receives pixel values representing the image data. Six batches of convolutional layers are then applied, each increasing the number of filters to extract intricate spatial features. By utilising a set of learnable filters, these layers can effectively detect patterns and features at different spatial locations within the image, regardless of their exact positions. This ability is essential for image classification tasks as it allows the network to learn meaningful representations of the input data, capturing important visual cues such as edges, textures, and shapes associated with different flower species. A primary concern when training was preventing overfitting due to the complex and limited nature of the dataset. To address this, batch normalisation is applied to the convolutional layers to stabilise and expedite training by normalising activations. This technique ensures that the network learns more efficiently and effectively by reducing internal covariate shift, ultimately leading to faster convergence and improved generalisation[7]. Also, Rectified Linear Units (ReLU) are applied to each layer, introducing non-linearity, crucial for capturing complex patterns in the data. A max-pooling layer, with a 2x2 kernel, is then applied after each batch to facilitate spatial dimension reduction, mitigating computational burden and the risk of overfitting [8]. While dropout is not usually used between pooling and convolutional layers, it can improve accuracy, as observed [9]. After the six batches, the features are flattened, and a fully connected layer maps the aggregated features from the convolutional layers to the 102 output classes, enabling accurate classification.

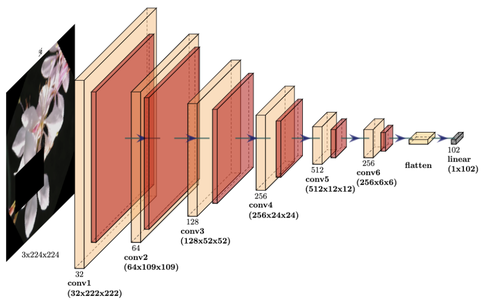

Fig. 1. CNN Architecture

B. Data Preprocessing and Augmentation

Data augmentation involves applying various transformations such as random resized crops, flips, colour jitter, rotations, affine transformations, and perspective changes to each image. A custom transform combines these transformations randomly. The data is then normalised using mean and standard deviation calculated from the training set. This approach significantly improves performance by diversifying the dataset with more combinations of transformations.

C. Training Procedure

1) Loss Function: The network utilizes Cross-entropy Loss to measure performance, defined as:

$$L(y, \hat{y}) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{C} y_{ij}\log(\hat{y}_{ij})$$

where N is the number of samples, C is the number of classes, y_{ij} is a binary indicator (0 or 1) if class j is the correct classification for sample i, and $\hat{y}_{ij}$ is the predicted probability of sample i belonging to class j.

Cross-entropy loss effectively measures the difference between predicted and actual class probabilities, guiding the model towards better predictions. It's well-suited for multi-class tasks, offering a nuanced measure of accuracy by considering entire probability distributions. By penalising misclassifications more heavily, it encourages accurate differentiation between classes, crucial in scenarios with numerous classes and limited data [10]. It's suitable for probabilistic models like neural networks, aiding in learning meaningful representations and making reliable predictions.

2) Optimizer: The low learning rate (0.0001) was selected through hyperparameter tuning for stable training and precise parameter adjustments to prevent overfitting. The Adam optimizer, blending AdaGrad and RMSProp features, effectively adjusted learning rates based on gradient moments, promoting faster convergence [11]. Adam's momentum maintains update directionality, even with small gradients, accelerating optimisation. Alongside a learning rate scheduler, adjusting rates by 0.1 after 10 epochs without improvement, Adam optimised training, enhancing model performance and generalisation.

3) Hyperparameters: The model undergoes 500 epochs of training, each batch comprising 16 samples. Early stopping is invoked after 15 epochs to prevent overfitting. After testing, a patience of 15 seemed to work best, giving the model time to recover from potential stagnation points. The decision to train for 500 epochs serves as a safety net, but the model typically trains for approximately 200 epochs.

III. Results and Evaluation

TABLE I
Hyperparamter Tuning

No.	Time Taken	Accuracy	Precision	F1-Score
1	15.93	0.4113	0.5108	0.4112
2	27.67	0.5394	0.6015	0.5420
3	48.14	0.5777	0.6338	0.5786
4	33.91	0.5637	0.6315	0.5684
5	35.70	0.5560	0.6155	0.5580
6	42.47	0.5822	0.6245	0.5821
7	57.52	0.6003	0.6405	0.6010
8	62.32	0.6284	0.6712	0.6317
9	56.44	0.6848	0.7204	0.6838

The initial model, a baseline CNN with a batch size (BS) of 16, LR of 0.0001, and early stopping at 8 epochs, achieved an accuracy of 58.72% after 40.67 minutes. A series of experiments were conducted to tune hyperparameters and assess their impact on model performance. Table I summarises the accuracy, precision, and F1-score for each experimental configuration. These evaluation metrics provide a balanced view of model performance across the 102 different classes, offering insights into the model's ability to make reliable predictions and distinguish between flower species. Experimenting with different setups revealed insightful trends: larger batch sizes, such as 32 (2), led to a reduction in accuracy to 53.94%, underscoring the effectiveness of smaller batches. Efforts to deepen the CNN (3) with extra fully connected layers resulted in a significant accuracy decrease to 41.13%. Adjusting momentum (4) slightly upwards to 0.95,0.999 from the original model saw a minor accuracy drop to 57.77%. Introducing weight decay (WD) of 0.01 with AdamW optimiser (5) yielded an accuracy of 56.37%. The implementation of a learning rate (LR) scheduler, specifically ReduceLROnPlateau (6), led to a reduction in accuracy to 55.60%. Switching to the Stochastic Gradient Descent (SGD) optimiser with LR set to 0.001 (7) resulted in an accuracy of 58.22%, while increasing the patience to 15 epochs with SGD (8) improved accuracy to 60.03%. Fine-tuning Adam optimiser with the LR Scheduler and longer patience (9) achieved the highest accuracy recorded at 62.84%. Ultimately, the most significant accuracy increase was observed with extreme data augmentation techniques (10), achieving an accuracy of 68.48% on the test set.

The model was trained on Google Colab using an L4 GPU. The duration of training for each model configuration is reported in I, providing insight into the computational resources required. The training process for the final model took approximately 62.32 minutes to complete. The environment used PyTorch version 2.0 for deep learning framework dependencies.

IV. Conclusion

Despite challenges like the dataset's small size and limited computational resources, the final model achieved a commendable 62.84% accuracy. This success stemmed from extensive hyperparameter tuning and effective data augmentation techniques, crucial for generalising from limited training data. Smaller learning rates and batch sizes proved effective, aligning with studies suggesting they improve generalization in deep neural networks [12]. The model's responsiveness to these adjustments suggests potential in data-constrained scenarios. However, certain classes, such as class 42, posed challenges due to less defined features. Improving the model's ability to capture subtle distinctions in flower morphology is crucial. Layer visualisation techniques could provide insights into the model's interpretation of image features, guiding enhancements [13]. Incorporating pre-trained models or exploring advanced architectures might capture complex patterns without expanding the dataset. For classes like 42, targeted data augmentation or manual enhancement of training samples could boost accuracy. Expanding the dataset could also improve performance, as larger datasets generally enhance neural network robustness and accuracy [14]. Pre-trained models on large datasets can leverage learned features, enhancing performance [15]. Implementing these strategies, alongside training adjustments, could yield a more accurate system capable of distinguishing nuanced differences in flower species.

References

[1] D. S. Shakya, "Analysis of artificial intelligence based image classification techniques," *Journal of Innovative Image Processing*, vol. 2, pp. 44–54, Mar. 2020. DOI: 10.36548/jiip.2020.1.005.

[2] J. Wäldchen, M. Rzanny, M. Seeland, and P. Mäder, "Automated plant species identification—trends and future directions," *PLOS Computational Biology*, vol. 14, A. Bucksch, Ed., e1005993, Apr. 2018. DOI: 10.1371/journal.pcbi.1005993.

[3] D. Lowe, *Object recognition from local scale-invariant features*, 1999. [Online]. Available: https://www.cs.ubc.ca/~lowe/papers/iccv99.pdf (visited on 05/12/2024).

[4] D. Andina, A. Voulodimos, N. Doulamis, A. Doulamis, and E. Protopapadakis, "Deep learning for computer vision: A brief review," *Computational Intelligence and Neuroscience*, vol. 2018, p. 7 068 349, 2018. DOI: 10.1155/2018/7068349. [Online]. Available: https://doi.org/10.1155/2018/7068349.

[5] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The Bulletin of Mathematical Biophysics*, vol. 5, pp. 115–133, Dec. 1943. DOI: 10.1007/bf02478259. [Online]. Available: https://link.springer.com/article/10.1007/BF02478259.

[6] M.-E. Nilsback and A. Zisserman, *102 category flower dataset*, www.robots.ox.ac.uk, 2008. [Online]. Available: https://www.robots.ox.ac.uk/~vgg/data/flowers/102/ (visited on 05/15/2024).

[7] S. Ioffe and C. Szegedy, *Batch normalization: Accelerating deep network training by reducing internal covariate shift*, arXiv.org, 2015. [Online]. Available: https://arxiv.org/abs/1502.03167 (visited on 05/16/2024).

[8] S. Sun, B. Hu, Z. Yu, and X. Song, "A stochastic max pooling strategy for convolutional neural network trained by noisy samples," *International Journal of Computers Communications Control*, vol. 15, Feb. 2020. DOI: 10.15837/ijccc.2020.1.3712. (visited on 02/20/2020).

[9] H. Wu and X. Gu, "Towards dropout training for convolutional neural networks," *Neural Networks*, vol. 71, pp. 1–10, Nov. 2015. DOI: 10.1016/j.neunet.2015.07.007. [Online]. Available: https://cs.nju.edu.cn/wujx/paper/CNN.pdf (visited on 05/17/2024).

[10] B. Barz and J. Denzler, *Deep learning on small datasets without pre-training using cosine loss*, arXiv.org, Dec. 2019. DOI: 10.48550/arXiv.1901.09054. [Online]. Available: https://arxiv.org/abs/1901.09054 (visited on 05/08/2024).

[11] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, arXiv.org, Dec. 2014. [Online]. Available: https://arxiv.org/abs/1412.6980.

[12] *A Bayesian Perspective on Generalization and Stochastic Gradient Descent*, ArXiv, Cornell University, Jan. 2017. DOI: 10.48550/arxiv.1710.06451. (visited on 05/17/2024).

[13] M. D. Zeiler and R. Fergus, *Visualizing and understanding convolutional networks*, Nov. 2013. DOI: 10.48550/arxiv.1311.2901.

[14] H. L. Dawson, O. Dubrule, and C. M. John, "Impact of dataset size and convolutional neural network architecture on transfer learning for carbonate rock classification," *Computers Geosciences*, vol. 171, p. 105 284, 2023. DOI: https://doi.org/10.1016/j.cageo.2022.105284. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0098300422002333.

[15] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, *How transferable are features in deep neural networks?* Neural Information Processing Systems, 2014. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2014/hash/375c71349b295fbe2dcdca9206f20a06-Abstract.html (visited on 05/17/2024).